情報処理学会論文誌　数理モデル化と応用　Vol.7　No.2　1–16 (Nov. 2014)

Compact Genetic Algorithmを導入した学習分類子システムによる分類子数の削減

中田 雅也[1,2,a)]　ピエール・ルカ・ランチ[3]　田島 友祐[1]　高玉 圭樹[1]

受付日 2013年8月26日，再受付日 2013年10月19日，
採録日 2013年11月22日

概要：本論文では，学習分類子システム（Learning Classifier System: LCS）において，学習する分類子数を削減するために，Compact Genetic Algorithm を用いた確率モデル型分類子生成法を提案する．提案分類子生成法は，1) 分類子が持つ部分解の存在確率をモデル化することで不要な分類子の生成を抑制し，2) 従来の確率モデル型分類子生成法が適用困難であった強化学習問題クラスに適用可能である．教師あり学習問題（Multiplexer 問題）と強化学習問題（Grid world 問題）において，提案分類子生成法を導入したLCS を適用したところ，次の知見を得た．まず，1) 提案 LCS は従来 LCS よりも，少ない学習回数で最適解を学習可能であり，2) 従来 LCS が学習した分類子数に対し，提案 LCS は最小でも 49%（最大で 76%）削減した分類子数で学習可能であることを示した．したがって，提案分類子生成法は，最適解を持つ分類子を早期に生成可能であり，不要な分類子の生成を抑制可能であることを示した．

キーワード：学習分類子システム，遺伝的アルゴリズム，確率モデル型遺伝的アルゴリズム

Rule Reduction in Learning Classifier System Using Compact Genetic Algorithm

Masaya Nakata[1,2,a)]　Pier Luca Lanzi[3]　Yusuke Tajima[1]　Keiki Takadama[1]

Received: August 26, 2013, Revised: October 19, 2013,
Accepted: November 22, 2013

Abstract: This paper proposes a novel probability model based rule-discovery mechanism using Compact Genetic Algorithm for Learning Classifier System (LCS), which evolves classifiers based on an extracted attribute of classifier conditions, to reduce a size of classifiers are needed in LCS. The proposed rule-discovery mechanism can 1) generate good classifiers that conditions have good building blocks; and 2) solve both single-step problems and multi-step problems where conventional probability-model based rule discovery mechanisms are hard to be applied. This paper applies LCS with the proposed rule-discovery mechanism to both a single-step problem (the multiplexer problem) and a multi-step problem (the grid world problem). Experimental results show following implications: 1) the proposed LCS can reach optimal performances faster than a conventional LCS; and 2) it can reduce the size of classifiers by at least 49% of that of the conventional LCS. Our conclusion is that the proposed rule-discovery mechanism can generate optimal classifiers with fewer generations than the conventional rule-discovery mechanism, and that it can control generating inaccurate classifiers toward the rule reduction.

Keywords: learning classifier system, genetic algorithm, compact genetic algorithm

[1] 電気通信大学大学院
Graduate School the University of Electro-Communications, Chofu, Tokyo 182–0021, Japan
[2] 日本学術振興会特別研究員（DC1）
Research Fellow of Japan Society for the Promotion of Science (DC1), Chiyoda, Tokyo 102–0083, Japan
[3] ミラノ工科大学
Politecnico di Milano, Milano, Italy
[a)] m.nakata@cas.hc.uec.ac.jp

1. はじめに

学習分類子システム（Learning Classifier System: LCS）[15] は，機械学習と進化計算を組み合わせた環境適応システムであり，次の特徴を有することから，有用な知識獲得システムとして注目されている．まず，LCS は，1) 簡易なルールである分類子を知識として学習すること

で，高精度かつ解釈性に優れた知識獲得技術 [3], [32], [33] を有する．加えて，2) 教師あり学習（Supervised Learning）や強化学習（Reinforcement Learning: RL）[29], [30] が扱う幅広い機械学習問題に適用可能である．たとえば，LCS はクラス分類問題 [1], [4], [10], [18] や，クラスタリング問題 [27], [31]，パターン認識問題 [11], [25]，ロボット行動獲得問題 [16], [17] 等に適用されている．ここで，LCS における知識獲得技術は一般化（generalization）と呼ばれ，複数の環境状態に照合する汎用的な分類子を学習することで実現される．

現在主流の LCS は，LCS を構成する 2 種類の機械学習手法（教師あり学習と強化学習）に応じて分類される．まず，教師あり学習を用いた LCS である Supervised Learning-based LCS（UCS）[3] は，環境適応に最低限必要な分類子のみ学習するため [20]，教師あり学習問題においては少ない分類子数で学習可能である [3]．しかし，UCS は教師データが設計不可能な未知環境を扱う強化学習問題に適用できない．一方で，強化学習を用いた LCS である Accuracy-based LCS（XCS）[9], [38] は，LCS が扱う全問題クラス（教師あり学習問題と強化学習問題）に適用可能である．しかし，XCS は全状態行動空間を網羅的に学習するため [9]，膨大な分類子数を必要とするという問題がある [20]．

本論文では，上記の従来 LCS の限界を克服するために，少ない分類子数で全問題クラスに適用可能な LCS の構築を目指す．このために，本論文は XCS 内の分類子生成法（Rule-discovery mechanism）に着目し，不要な分類子の生成を抑制することで，XCS における分類子数を削減することを考える．ここで，分類子生成法は進化計算によって実現され，一般的に遺伝的アルゴリズム（Genetic Algorithm: GA）[13] が用いられる [3], [37], [38]．特に，XCS では，学習不十分な分類子が削除されることを防ぐため，毎世代に 2 つの子個体（分類子）のみ生成（2 個体のみ削除）する定常状態 GA（Steady-State GA）[36] が用いられる [9]．しかし，定常状態 GA は，毎世代に 2 つの親個体のみ考慮して子個体を生成するため，正しい分類子を獲得するまでに，多様な分類子を生成する．その結果，XCS は，全状態行動空間における最適解を獲得するために，膨大な分類子数を生成する必要がある．

そこで，我々は，分布推定アルゴリズム（Estimation of Distribution Algorithm: EDA）[22], [24] の一手法である Compact Genetic Algorithm（cGA）[14] を用いた確率モデル型分類子生成法を提案する．提案手法は，親個体が持つ部分解の存在確率を確率ベクトルでモデル化することで，有望な部分解を持たない分類子（不要な分類子）の生成を抑制可能とする．したがって，提案手法は，複数の親個体を考慮して子個体を生成する点で定常状態 GA と異なる．加えて，提案手法は，XCS のための分類子生成法であるため，従来 LCS（Compact Classifier System: CCS [23], Attribute-Feedback UCS: AF-UCS [35]）の確率モデル型分類子生成法が適用困難であった強化学習問題に適用可能である．具体的には，CCS や AF-UCS の分類子生成法は，評価関数もしくは教師データを用いて確率モデルを生成するため，XCS に適用不能なメカニズムであるが，提案手法では，教師データ等を用いずに確率モデルを生成するメカニズムを持つ．提案手法の有効性を検証するために，本論文では，提案手法を導入した XCS を教師あり学習問題（Multiplexer 問題 [38]）と強化学習問題（Grid world 問題 [7]）に適用する．

本論文の構成は次のとおりである．以下，2 章では，CCS や AF-UCS 等の関連研究について説明する．3，4 章では，提案手法が基にするシステムである XCS と cGA のメカニズムについて述べ，5 章で提案手法について説明する．6，7 章では，Multiplexer 問題と Grid world 問題における実験結果を示し，提案手法の有効性について評価する．8 章では，提案手法のメカニズムの挙動を解析するために追加実験を行う．最後に，9 章で本論文をまとめる．

2. 関連研究

本章では，1) 分類子数の削減手法ならびに 2) 確率モデル型分類子生成法に関する既存研究について述べる．

2.1 従来の分類子数削減手法

LCS が膨大な分類子数を必要とすることで次の問題が生じる．まず，1) ある環境状態に照合する分類子が複数存在することで，冗長な知識が生成され解釈性が低下する．さらに，2) 不要な分類子も学習するため多大な学習回数を要する．XCS における分類子数削減手法の従来手法として，学習終了後の分類子数を圧縮する方法（rule compaction）が提案されている．

一般的に学習終了後の分類子集団には，学習後期に生成されたことで十分に学習されていない分類子や，誤って一般化された分類子といった不要な分類子が多数含まれる [12]．そこで分類子数圧縮方法 [12], [39], [40] は，学習終了後の分類子集団において，上記の不要な分類子を削除し，正しく一般化された分類子のみ抽出することで，分類子数を削減する．しかしながら，一般的に同手法は，学習性能の低下を許容して分類子数を圧縮する方法 [12] で実現されるため，XCS の学習性能を維持しつつ分類子数を圧縮することが困難である．異なる方法として，XCS が学習すべき分類子数を削減することで分類子数を圧縮する方法 [40] は，学習性能の低下を防ぐことが可能であるが，XCS よりも獲得する知識数が減少するという問題がある．加えて，学習中の XCS は，依然として膨大な分類子数が必要であり，最適解を得るまでに多大な学習回数を要するという問題（上記問題 2）は解決されていない．

2.2　従来の確率モデル型分類子生成法

従来の確率モデル型分類子生成法として，Compact Classifier System（CCS）[23] と Attribute-Feedback UCS（AF-UCS）[34] 内で同分類子生成法が提案されている．提案手法と同様に確率ベクトルを用いて分類子を生成するが，強化学習問題に適用困難であるという問題が存在する．以下に，これらの手法について説明する．

2.2.1　Compact Classifier System（CCS）

LCS は，XCS や UCS が属するミシガン型 LCS（Michigan-style LCS）[15] とピッツバーグ型 LCS（Pittsburgh-style LCS）[28] に分類される．ミシガン型 LCS はオンライン学習を用いて 1 つの分類子集団を学習することに対し，ピッツバーグ型 LCS はオフライン学習を用いて複数の分類子集団を学習する点が特徴である [19]．

CCS はピッツバーグ型 LCS を基にした LCS であり，複数の分類子集団の代わりに複数の確率ベクトルを保有する．CCS における確率モデル型分類子生成法は，各確率ベクトルから子個体を生成するが，その子個体の適応度に応じて確率ベクトルを更新することで，良好な確率モデルを構築する．

したがって，CCS の特徴は，分類子集団とそれらを構成する分類子を保有する必要がないことである．しかしながら，CCS は，ピッツバーグ型 LCS の特性から，オンライン学習問題（強化学習問題）に適用することができない．加えて，CCS の確率モデル型分類子生成法についても，子個体の適応度が算出可能である（評価関数が設計可能である）ことを前提としているため，評価関数が設計困難な強化学習問題に適用することができない．

2.2.2　Attribute-Feedback UCS（AF-UCS）

AF-UCS は，UCS に確率モデル型分類子生成法を導入した LCS であり，雑音を含むエピスタシス性が強い実データにおける知識獲得システムとして用いられる [35]．このような複雑な実データから知識を獲得するために，AF-UCS は環境状態（データ）ごとに存在する知識の部分解を確率モデルを用いて獲得する．AF-UCS における確率モデル型分類子生成法は，CCS と同様に確率ベクトルから子個体を生成し，確率ベクトルを更新する．加えて，同分類子生成法は，UCS を用いることで，教師データから判別した正しい分類則を持つ分類子から確率ベクトルを更新する．また同分類子生成法は，確率ベクトルを基に交叉と突然変異を行うことで，確率ベクトルより抽出した部分解を分類子に反映し，有望な部分解を持つ分類子の生成を促進する．

しかしながら，AF-UCS が用いる UCS は強化学習問題に適用不可能であることに加えて，AF-UCS の確率モデル型分類子生成法には次の問題が存在する．まず，1) 同手法は教師データを用いて正しい確率モデルを構築するため，強化学習問題では良好な確率モデルを構築できない．さらに，2) 環境状態数だけ確率ベクトルが必要であることから，広大な状態空間を持つ問題に適用限界がある．

3.　Accuracy-based LCS（XCS）

提案手法である cGA を用いた確率モデル型分類子生成法は，XCS 内で用いる分類子生成法であり，XCS における分類子の評価値やメカニズムを用いて分類子を生成する．本章では，XCS で用いられる分類子の構成と XCS のメカニズムについて説明する．

3.1　分類子

XCS で用いられる分類子（classifier）は IF（条件）-THEN（行動）ルールで表現され，環境状態と照合する条件部（condition: C）と，照合後に実行する行動部（action: A）からなる．一般的に条件部と行動部は 0，1 のビット列で形成されるが，特に条件部は任意の値を示す#（don't care）を組み込むことで，複数の環境状態に適応可能な汎用的な条件部が形成される．各分類子は予測値（prediction），誤差値（error），適応度（fitness）および重合度（numerosity）の評価値を持つ．ここで重合度は，条件部と行動部がともに等しい分類子をマクロ分類子（macro-classifier）として集約した数を意味する．分類子の上限数（max population size）をパラメータ N と定め，j 番目の分類子 cl_j の予測値，誤差値，適応度，重合度をそれぞれ p_j，ϵ_j，F_j，num_j と記す．

3.2　メカニズム

図 1 に，XCS のメカニズムの概要を示す．XCS のメカニズムは実行部，強化部および発見部から構成され，効率良く一般化を行うための包摂（subsumption）と呼ばれる操作が存在する．なお，分類子生成法は発見部で用いられる．XCS のメカニズムは，まず実行部が適用され，強化部，発見部の順で実行される．ここで，包摂は実行部内で適用される．実行部では，入力状態に対する行動を選択し実行する過程を担う．強化部では，実行した行動の選択に寄与した分類子の各評価値を更新する過程を担う．最後に，発見部では，GA による分類子の生成と削除を通して，正しい部分解を持つ分類子を探索する過程を担う．

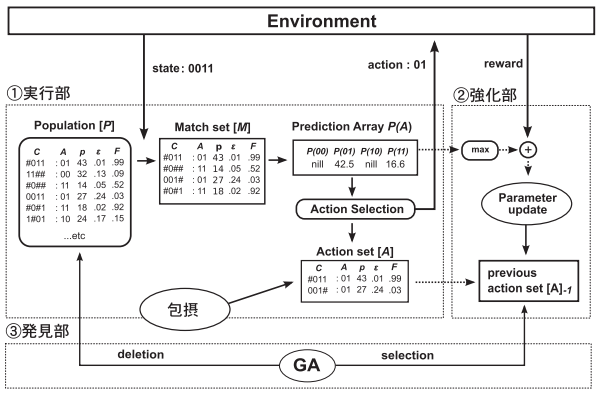

図 1　XCS のメカニズムの概要 [38]
Fig. 1　Overview of XCS.

以下に，各メカニズムならびに包摂について説明する．

3.2.1　実行部

実行部は分類子集団（population: $[P]$）の中から環境状態と照合する分類子の行動を選択し出力する処理を担う．環境より入力される状態（state）は，一般的に 0，1 のビット列からなり，$[P]$ 内の各分類子の条件部と照合され，照合した分類子は照合集合（match set: $[M]$）を形成する．このとき，$[M]$ 内に存在する分類子が持つ行動部の種類の数がパラメータ θ_{mna} の値未満の場合，条件部が入力状態に照合し，かつ，$[M]$ 内に存在しない行動を持つ分類子を θ_{mna} の値以上に上回るまで生成する．この操作により，$[P]$ 内に存在しない分類則（行動）を持つ分類子を生成することで，正しい分類則を持つ分類子を効率良く探索可能となる．ここで，θ_{mna} は $[M]$ 内に存在すべき行動種類数の最小値（Minimal Number of Actions）を意味し，被覆により生成される分類子の条件部，行動部ならびに各評価値は次のように設定される [9]．まず，条件部は，入力状態と同一に設定されるが，各ビットに対して確率 $P_\#$ で#に置き換えられる．行動部は，$[M]$ 内に存在しない行動からランダムに選択した行動を設定する．最後に，分類子の予測値 p，誤差値 ϵ ならびに適合度 F は初期値 p_I，ϵ_I，F_I に設定し，重合度 num を 1 に設定する[*1]．

$[M]$ を形成後，式 (1) に従って各行動の予測報酬を計算し予測配列（prediction array: $P(a)$）に記憶し，これを選択確率として行動選択を行う．ただし，式中の $[M]\,|_{a_i}$ は，$[M]$ 内で行動部に行動 a_i を持つ分類子の集合であり，$[M]$ 内に存在しない行動の予測報酬は nill とされ行動選択の対象とならない．たとえば，図 1 において，行動 00 を行動部に持つ分類子が $[M]$ 内に存在しないため，予測報酬 $P(00)$ は nill となり行動選択の対象とならない．一方で，行動 01 に対する予測報酬は，$P(01) = (43 \times 0.99 + 27 \times 0.03)/(0.99 + 0.03) = 42.5$ と算出される．最後に，照合集合から選択された行動を行動部に持つ分類子をまとめて行動集合（action set: $[A]$）を形成し，$[A]$ 内の分類子に対して後述する包摂の操作が適用される．その後，選択された行動が実行される．この一連の処理単位をステップと呼ぶ．

$$P(a_i) = \frac{\sum_{cl_k \in [M]|a_i} p_k \times F_k}{\sum_{cl_k \in [M]|a_i} F_k} \tag{1}$$

3.2.2　強化部

強化部は実行部の処理が完了後，環境から得られた報酬 r に基づいて $[A]$ 内の各分類子の予測値，誤差値および適応度を式 (2)〜(4) のように更新する．

$$P \leftarrow r + \gamma \, max\, P(a) \tag{2}$$

$$p_j \leftarrow p_j + \beta(P - p_j) \tag{3}$$

$$\epsilon_j \leftarrow \epsilon_j + \beta(|P - p_j| - \epsilon_j) \tag{4}$$

[*1] p_I，ϵ_I，F_I は 0 に近い値に設定することが好ましく [9]，本論文では各値を 0.01 と設定する．

P は予測値 p_j を更新する際の目標値であり，前ステップの報酬 r と予測配列中の最大予測報酬 $maxP(a)$ を用いて計算される．パラメータ β $(0 \le \beta \le 1)$，γ $(0 \le \gamma \le 1)$ はそれぞれ学習率（learning rate），割引率（discount factor）と呼ばれ，それぞれ学習の更新速度と将来の報酬を考慮する度合いを制御する．ここで，強化学習問題では式 (5) に示すように，適応度に関する勾配を用いた予測値の更新式が用いられ，強化学習問題で正しく予測値を見積ることが可能である [7]．したがって，本論文では，教師あり学習問題では式 (3) を用いるが，強化学習問題では式 (5) を用いて予測値を更新する．

$$p_j \leftarrow p_j + \beta(P - p_j) \times \frac{F_j}{\sum_{cl_k \in [A]} F_k} \tag{5}$$

そして，更新後の誤差値 ϵ_j に基づき分類子の絶対的な正確さ κ_j と相対的な正確さ κ'_j が式 (6)，(7) のように計算される．最後に適応度 F_j が式 (8) に示すように更新され，強化部における一連の処理が完了する．ここで，ϵ_0 および ν は誤差 ϵ_j から絶対的な正確さ κ_j を計算する際のパラメータであり問題に応じて適切に設定する．

$$\kappa_j = \begin{cases} 1 & \text{if } \epsilon_j \le \epsilon_0 \\ \alpha(\epsilon_j/\epsilon_0)^{-\nu} & \text{otherwise.} \end{cases} \tag{6}$$

$$\kappa'_j = \frac{(\kappa_j \times num_j)}{\sum_{cl_k \in [\mathrm{A}]_{-1}} (\kappa_k \times num_k)} \tag{7}$$

$$F_j \leftarrow F_j + \beta(\kappa'_j - F_j) \tag{8}$$

3.2.3　発見部

発見部では定常状態 GA による分類子の生成と削除を通して，分類子集団 $[P]$ を進化させる．したがって，XCS で用いる分類子生成法は，定常状態 GA を用いた方法である．具体的には，前ステップ行動集合 $[\mathrm{A}]_{-1}$ を形成する各分類子が，それぞれ前回 GA の進化対象に設定されたときから経過したステップ数の平均値がパラメータ θ_{GA} の値を上回る場合に GA が実行される．

まず発見部では，GA に基づく分類子生成法が適用される．具体的には，$[\mathrm{A}]_{-1}$ 内の各分類子の適応度を用いたトーナメント選択（トーナメントサイズ τ）[6] より 2 つの分類子が親個体として選択される．次に，子個体として，それぞれの親個体の分類子と同様の条件部，行動部および各評価値を持つ分類子が 2 つ生成され，交叉（crossover）および突然変異（mutation）がそれぞれパラメータ χ，μ の確率で適用される．交叉が適用された場合，各子個体の各評価値は，2 つの親個体の評価値の平均値を設定する [9]．特に，XCS で用いられる突然変異（niche mutation）[9] では，分類子の条件部を#もしくは入力状態の属性値に変異させる．

最後に，子個体である 2 つの分類子は分類子集団に追加される．このとき，$[P]$ 内の分類子数がパラメータ N を

超えた場合，適応度の低い分類子が削除（deletion）され，N を超えないときは削除されない．ここで分類子の削除とは，削除する数だけ分類子の重合度を減少させることであり，重合度が 0 になると分類子が消滅する．

3.2.4　包摂

包摂は分類子集団内で，ある分類子と同じ行動部を持ち，かつ，より一般化された条件部を持つ分類子が存在した場合に前者を後者に統合することで，分類子条件部の一般化を促進する．また，統合後，後者の重合度は前者の重合度だけ加算され，前者の分類子は消滅する．たとえば，2 つの分類子 $cl_a = \{C{:}\#\#\#1,\ A{:}1,\ num{:}5\}$，$cl_b = \{C{:}0\#\#1,\ A{:}1,\ num{:}1\}$ が存在するとき，cl_a は cl_b よりも一般化された分類子であり，包摂の操作によって cl_a の重合度は 6 となり，cl_b は消滅する．包摂は一方の条件部を他方の条件部へと変更する操作であり，分類子集団内のマクロ分類子の数の減少に寄与する．

4.　Compact Genetic Algorithm（cGA）

本章では，提案手法が基にする分布推定アルゴリズムである Compact Genetic Algorithm（cGA）について説明する．具体的には，cGA における確率モデルの構築法ならびに個体の生成法について説明する．

cGA はビットストリング型の確率モデルを用いた GA である．cGA では，0，1 のバイナリで表現される環境状態を持つ環境を対象とし，GA における母集団の代わりに確率モデルを用いる．cGA における確率モデルは，遺伝子長と同じ長さの確率ベクトルを用い，環境内の最適解を確率ベクトルで表現する．ここで，各確率は対応する遺伝子座が 1 である確率を意味する．

まず，確率ベクトルにおける各確率は 0.5 として初期設定される．そして，確率ベクトルから子個体を生成するが，子個体の各遺伝子座の属性値（0 もしくは 1）は対応する確率より決定される．cGA では，2 つの子個体を生成後，評価関数より子個体を評価し適応度（fitness）を得る．その後，適応度が高い子個体を winner，低い子個体を loser と定め，両者の各遺伝子座の属性値の違いを用いて，確率ベクトルを更新する．具体的には，各遺伝子座において，winner と loser が異なる属性を持つ場合のみ，対応する確率を更新する．winner が 1 で loser が 0 である場合，確率の値を更新率（$1/n$）だけ増加させる．一方で，winner が 0 で loser が 1 である場合，確率の値を $1/n$ だけ減少させる．なお，同一の属性値を持つ場合，対応する確率は更新されない．ここで，n は任意に設定可能であるが，母集団上限数 n に設定した GA を cGA でシュミレーションすることができる [14]．

5.　提案手法

本章では，提案手法である cGA を基にした確率モデル型分類子生成法について述べ，関連研究に対する提案手法の位置づけについて述べる．

提案手法は，cGA を基にした確率モデル型分類子生成法であり，子個体を確率ベクトルより生成する．提案手法の特徴は，1) XCS の分類子生成法（定常状態 GA）と比較して，複数の親個体から子個体を生成すること，2) 従来の確率モデル型分類子生成法と比較して，強化学習問題に適用できることである．cGA との差異は，i) winner と loser を判別せずに 2 つ以上の個体から確率を計算すること，ii) 確率ベクトルを保有ならびに更新せず毎世代確率ベクトルを構築すること，さらに，iii) 生成した子個体に対し，確率ベクトル用いた突然変異を適用することである．そして，XCS で用いる GA を用いた分類子生成法との差異は，毎世代に複数の分類子を考慮して，子個体を生成することである．このため，提案手法は，a) 複数個体からの確率ベクトルの生成（上記 i と ii に相当），b) 確率ベクトルに基づく突然変異（上記 iii に相当）の 2 つのメカニズムにより構成される．以下，各メカニズムについて説明する．

5.1　複数個体からの確率ベクトルの生成

5.1.1　概要

cGA は，子個体の正しい適応度が評価関数より獲得できることを前提としてる．そのため，cGA は 2 つの個体を比較することで，winner と loser に正しく判別可能であり，正しい部分解（属性値）を識別できる．しかしながら，提案手法は，強化学習問題に適用するために，評価関数や教師データを用いずに正しい部分解を識別することが求められる．しかし，XCS より学習した分類子の適応度は，学習途中では必ずしも正しい値に収束しているとは限らない．これは，XCS に導入する提案手法では，winner と loser が一意に決定できないことを意味する．そのため，提案手法では，複数の分類子に存在する部分解の存在確率を算出し，存在確率の高い部分解を有望な部分解として扱うことを考える．

5.1.2　メカニズム

図 2 に複数個体から確率ベクトルを生成する過程の概要図を示す．提案手法は，XCS と同様に行動集合 [A] 内の分類子を基に分類子を生成する．一般的に，適応度 F が高い

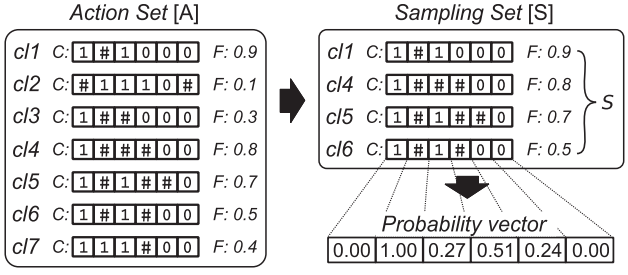

図 2　複数個体からの確率ベクトルの生成の概要図
Fig. 2　Calculation of probability vector based on sampling set.

分類子は有望な部分解を持つ可能性が高いが，[A] には適応度が低い分類子が多数存在している．提案手法では，適応度が低い分類子が持つ誤った部分解を抽出することを防ぐため，高い適応度を持つ分類子から構成される標本集合（sampling set）[S] を生成する．

具体的には，[A] からトーナメント選択 [6] により選択された高い適合度を持つ分類子（親個体）を，[S] に格納する．このとき，選択する親個体の数を，[A] 内の分類子数の sr（sampling rate）%と定める．ここで，sr は選択する親個体の数を調整するパラメータであり，$0 < sr \leq 1.0$（100%）までの任意の値に設定する．たとえば，$sr = 0.3$ に設定した場合，分類子数が 10 である [A] から 3（$= 10 \times 0.3$）つの分類子を親個体として選択し，[S] に格納する．また，$sr = 1.0$ に設定した場合，[A] 内のすべての分類子を親個体にすることを意味し，[S] は [A] と同一の分類子により構成される．ただし，親個体はすでに選択した親個体を除く分類子から選択する．

[S] を生成後，[S] 内の分類子から確率ベクトルを計算する．提案手法では 0，1 のバイナリで表現される環境を扱うが，各確率は対応する遺伝子座が#（generalized bit）もしくは 0，1（specified bit）であるかを表現する．具体的には，各確率は#である確率を意味する[*2]．**図 3** に，具体的な確率ベクトルの算出アルゴリズムを示す．i 番目の遺伝子が#である確率 $prob_i$ は，[S] 内の分類子の条件部 C の対応する遺伝子座 C_i から算出される．このとき，各分類子の適応度 F の重みをつけて計算する．たとえば，図 2 において，[S] 内の各分類子の条件部の 1 番目の遺伝子座は，すべて specified bit（1）であり，確率ベクトル内の 1 番目の確率は 0 となる．また，3 番目の遺伝子座は，cl_4 のみが generalized bit（#）である．そのため，3 番目の確率は 0.27（$= 0.8/(0.9 + 0.8 + 0.7 + 0.5)$）と算出される．

その後，cGA と同様に確率ベクトルから 2 つの子個体を生成し，分類子集団 [P] に追加する．ここで，子個体の各遺伝子座の属性値は，対応する確率により generalized bit（#）もしくは specified bit（0，1）に設定される．specified bit となった場合は，入力状態 σ の対応する属性値 σ_i が設定される．そして，XCS の発見部と同様に，子個体に対して交叉および突然変異がそれぞれパラメータ χ，μ の確率で適用される．ただし，突然変異については後述する確率ベクトルに基づく突然変異が適用される．交叉が適用された場合，子個体の各評価値は，[S] 内の分類子の評価値の平均値を設定する．

[P] 内の分類子数がパラメータ N を超えた場合，適合度の低い分類子が削除（deletion）され，N を超えないときは削除されない．なお，提案手法は，XCS における分類子生成法（定常状態 GA）に代わる手法であり，XCS と同様にパラメータ θ_{GA} を用いて提案手法を実行するかどうか決定される．

```
1: for all classifier cl in [S] do
2:     for i = 0 to Length of C of cl do
3:         if C_i is equal to # then
4:             prob_i += cl.F
5:         end if
6:     end for
7:     FitnessSum += cl.F
8: end for
9: for i = 0 to Length of C do
10:    prob_i /= FitnessSum
11: end for
```

図 3　確率ベクトルの算出アルゴリズム

Fig. 3　Algorithm of calculating probability vector.

[*2] [S] 内の全分類子は照合集合 [M] に含まれる分類子から構成されるため，specified bit になる遺伝子座は対応する環境状態の属性値（0，1）と一致する．そのため，確率ベクトルでは 0，1 を識別する必要がない．

5.2　確率ベクトルを用いた突然変異

5.2.1　概要

学習途中では分類子は正しい部分解を持っておらず，確率ベクトル内の各確率は十分に収束しない．そのため，生成される子個体の条件部は，確率ベクトルが示唆する有望な部分解（属性値）が反映されていないことが考えられる．そこで提案手法では，確率ベクトルが示唆する部分解を子個体に反映させるために，確率ベクトルに基づく突然変異を導入する．具体的には，提案する突然変異は，各遺伝子座の属性値が確率ベクトルの示唆する属性値と異なる場合，示唆された属性値へ変異させることで，分類子の条件部を変異させる．

5.2.2　メカニズム

図 4 に提案する確率ベクトルに基づく突然変異のアルゴリズムを示す．提案する突然変異は，AF-UCS で用いられる突然変異 [35] と同様に，1) 通常の突然変異（niche mutation）と 2) 確率ベクトルに基づく突然変異について，確率 P_C で適用する突然変異を決定する [34]．たとえば，

```
1: for i = 0 to Length of C do
2:     if RandomNumber ≤ P_C then
3:         Run niche mutation
4:     else
5:         if RandomNumber ≤ μ then
6:             if RandomNumber ≤ prob_i in vector then
7:                 C_i ← #
8:             else
9:                 C_i ← σ_i
10:            end if
11:        end if
12:    end if
13: end for
```

図 4　確率ベクトルに基づく突然変異のアルゴリズム

Fig. 4　Algorithm of mutation based on probability vector.

$P_C = 0.0$ である場合，確率ベクトルに基づく突然変異がつねに実行される．実行する突然変異を決定後，分類子の条件部の各遺伝子座について，突然変異確率 μ で突然変異が実行される．AF-UCS は，確率ベクトル内の確率の最大値を考慮して分類子を変異させるが，提案手法は，確率ベクトル中の対応する確率 $prob_i$ に応じて，遺伝子座の属性値を#もしくは対応する入力状態の属性値 σ_i に変異させる．

5.3　提案手法の位置づけ

ここでは，関連研究である 1) 従来の分類子数削減法ならびに 2) 従来の確率モデル型分類子生成法について，それぞれの観点から提案手法の位置づけについて述べる．

5.3.1　従来の分類子数削減法における提案手法の位置づけ

提案手法は，分類子数圧縮法と比較して，学習中に分類子数を削減することが可能である．これは，提案手法は学習段階中に不要な分類子の生成を抑制するためである．その結果，提案手法は，上記問題 1 に取り組むとともに上記問題 2 を解決可能である点で，従来の分類子数削減手法と異なる．

5.3.2　従来の確率モデル型分類子生成法における提案手法の位置づけ

表 1 に，従来（CCS と AF-UCS）の確率モデル型分類子生成法と提案手法の特徴の違いを示す．CCS と AF-UCS は，環境に存在する複数の最適解を確率モデルで表現することを目的としている．したがって，これらの手法は複数の確率モデルを保有する必要があり，確率モデル内の確率を更新することで最適解を求める．

具体的には，CCS は確率ベクトルで分類子集団を表現するため，分類子および分類子集団を保有する必要がないが，個体の評価に評価関数が必要である．そのため，CCS は評価関数が設計可能な問題のみ適用可能である．一方で，AF-UCS は，学習した分類子を用いて確率ベクトルを更新するため，分類子集団を生成する必要がある．また，AF-UCS は教師データを用いて分類子を学習するため，教師あり学習問題にのみ適用可能である．ここで，教師データと評価関数の違いは，評価関数は分類子の部分解を評価するが，教師データは分類子が示す分類則（IF，THEN の組合せ）を評価する点である．したがって，CCS では評価関数により正しい部分解を識別することができる．一方で，AF-UCS では正しい分類則は判別可能であるが，正しい部分解を持つ分類子を進化的に探索する必要がある．

提案手法は，確率モデルから良好な分類子を生成し分類子数を削減することを目的としている（最適解は，XCS と同様に分類子を用いて表現する）．具体的には，提案手法は，環境状態が入力されるたびに，その状態に適した確率モデルを生成する．加えて，提案手法は，強化学習の枠組みにより学習された分類子を用いて確率モデルを構築する．したがって，提案手法の特徴は，1) 複数の確率モデルを保有し更新する必要がなく，1 つの確率モデルを毎回生成すること，2) 評価関数や教師データを必要としないことである．そのため，提案手法は，広大な状態空間を持つ問題や強化学習問題に適用可能である．

6.　教師あり学習問題における実験

本章では，教師あり学習問題における実験を通して，提案手法の有効性を評価する．具体的には，教師あり学習問題における LCS の一般的なベンチマーク問題として，Multiplexer 問題 [38] を用いて実験を行う．

6.1　Multiplexer 問題

Multiplexer 問題は，入力と出力との間の高い非線形性と，入力の適切な一般化により入出力関係を縮約できる構造を持つため，LCS の一般化能力の評価に用いられる [6], [8], [38]．Multiplexer 問題で与えられる入力であるビット長 l は $k + 2^k$ であり，k は任意の整数である．また，出力はビット列 $b_0 b_1 \cdots b_{l-1}$ の最初から k ビット（b_0〜b_{k-1}）を 10 進数に変換した値 d を加えた b_{k+d} ビットの値となる．たとえば，$k = 2$ としたときに与えられる入力 110001 のアドレスは $b_0\ b_1 \rightarrow 11$ を変換した値 $d = 3$ と定まり，b_{2+3} のビット値である 1 が出力となる．

ここで，教師あり学習問題として扱う Multiplexer 問題は，教師データとして入力に対して正しい出力を LCS に与えることが可能であるが，XCS では教師データを扱う機構を有していないため，XCS が実行した行動を報酬の有無により評価する [38]．具体的には，任意の入力状態（入力）に対して正しい行動（出力）を実行した場合は報酬として $r = 1{,}000$ を与え，間違った場合は 0 を与える．また，本論文では $k = 4, 5$ とした 20-，37-Multiplexer 問題を用いる．

表 1　確率モデル型分類子生成法の特徴の違い
Table 1　Previous and proposed probability model-based rule-discovery mechanisms.

	CCS [23]	AF-UCS [35]	提案手法
確率モデルの数	複数	複数	1
確率モデルの学習	更新	更新	しない（毎回生成）
分類子集団	確率ベクトルで表現	生成する	生成する
個体の評価（学習）方法	評価関数	教師あり学習	強化学習
問題適用範囲	評価関数が設計可能な問題	教師あり学習問題	教師あり/強化学習問題

6.2　実験設定

6.2.1　概要

提案手法の有効性を評価するために，提案手法を導入した XCS（XCScGA）と従来手法である XCS を Multiplexer 問題に適用し，学習性能を評価する．ここで XCScGA は，XCS の発見部を提案手法に置き換えた LCS である．また，先行研究 [8] と同様に，実験は学習段階と評価段階から構成され，交互に実行する．学習段階では，ランダムに行動選択することで，環境内の状態空間を十分に学習し，正しい分類子を探索し学習する．一方で評価段階では，予測配列中の最大予測報酬が示す行動を選択（greedy 行動選択）することで，後述する評価基準を算出する．また，評価段階では，発見部を適用せず新しい分類子を生成しない [21], [38]．

6.2.2　評価基準とパラメータ設定

評価基準は，1) 正答率（Performance），2) 分類子集団 [P] 中の分類子数（Population size）を用いる．正答率は高い値であるほど，正確に一般化された分類子を学習できていることを示している．分類子数は少ない値であるほど，分類子集団中に不要な分類子が少ないことを意味する．これらの値は，評価段階で新たに与えられた異なる入力の 20-Multiplexer 問題について，greedy 選択したときの解答から計算する．学習回数は 1 試行につき，20-Multiplexer 問題では 100,000 回，37-Multiplexer 問題では 500,000 回行う．このとき，各評価基準は学習回数 5,000 回ごとの移動平均で示し，試行数 10 回の平均をとる．

パラメータ設定は先行研究と同様に設定する [8]．具体的には，20-Multiplexer 問題における XCS のパラメータ設定は，$N = 2{,}000$，$\nu = 5$，$\beta = 0.2$，$\epsilon_0 = 10$，$\theta_{mna} = 2$，$\theta_{GA} = 25$，$\chi = 0.8$，$\mu = 0.04$，$P_\# = 0.5$，$\tau = 0.4$ とする．XCScGA は，$sr = 0.2$（20%），$P_C = 0.2$ と設定し，その他のパラメータは XCS と同様に設定する．37-Multiplexer 問題における XCS と XCScGA のパラメータ設定は，$N = 5{,}000$，$P_\# = 0.65$ と設定し，その他は 20-Multiplexer 問題と同様である．

6.3　実験結果

6.3.1　20-Multiplexer 問題

図 5，**図 6** に，20-Multiplexer 問題における XCS と XCScGA の正答率ならびに [P] 中の分類子数を示す．各図の縦軸は，それぞれ正答率（Performance），分類子数（Population size）であり，横軸は学習回数（Iterations）を示している．

両図より，XCS は学習回数 35,000 回程度で正答率が 1 に達しているが，XCScGA は 25,000 回程度で正答率が 1 に達しており，XCS よりも 10,000 回少ない学習回数で正しい分類子を学習できることが分かる．XCS の分類子数は，学習回数 10,000 回程度で最大値となり，学習回数の経過にともなって次第に減少している．一方で，XCScGA

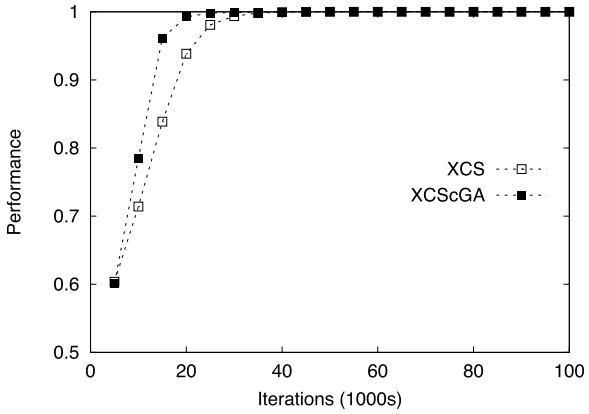

図 5　20-Multiplexer 問題における XCS と XCScGA の正答率
Fig. 5　Performances of XCS and XCScGA on 20-Multiplexer problem.

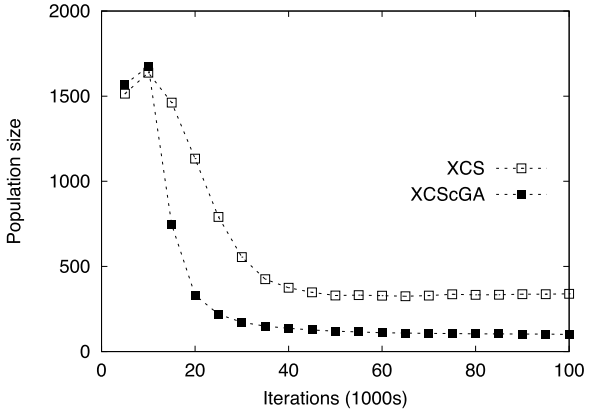

図 6　20-Multiplexer 問題における XCS と XCScGA の分類子数
Fig. 6　Population sizes of XCS and XCScGA on 20-Multiplexer problem.

は，XCS と同様に学習回数 10,000 回程度で最大値となるが，その後 XCS よりも急激に分類子数が減少している．これは，XCScGA の [P] 中の不要な分類子の数が減少していることを意味する．また，学習終了後の XCS の分類子数は 339 であるの対し，XCScGA の分類子数は 101 であることから，XCScGA は XCS の分類子数を約 70.5%削減していることが分かる．

6.3.2　37-Multiplexer 問題

図 7，**図 8** に，37-Multiplexer 問題における XCS と XCScGA の正答率ならびに [P] 内の分類子数を示す．

両図より，XCS は学習回数 350,000 回程度で正答率が 1 に達しているが，XCScGA は 100,000 回程度で正答率が 1 に達しており，きわめて少ない学習回数で正しい分類子を学習できている．XCS の分類子数については，学習回数 10,000 回程度で最大値となり，学習回数の経過にともなって次第に減少している．一方で，XCScGA は，学習回数 10,000～35,000 回の間は XCS よりも大きい値であるが，その後急激に分類子数が減少している．これは，XCScGA が，学習初期では XCS よりも多様な分類子を生成し大域

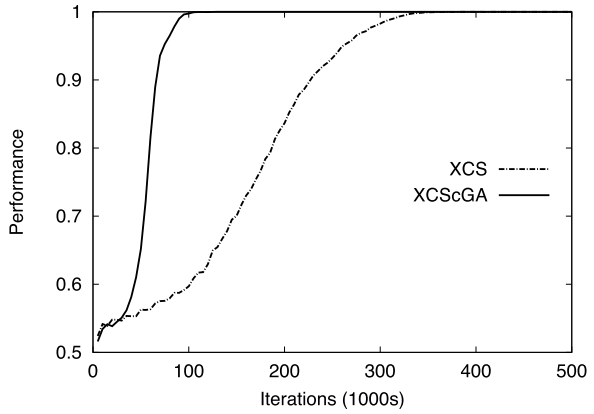

図 7　37-Multiplexer 問題における XCS と XCScGA の正答率
Fig. 7　Performances of XCS and XCScGA on 37-Multiplexer problem.

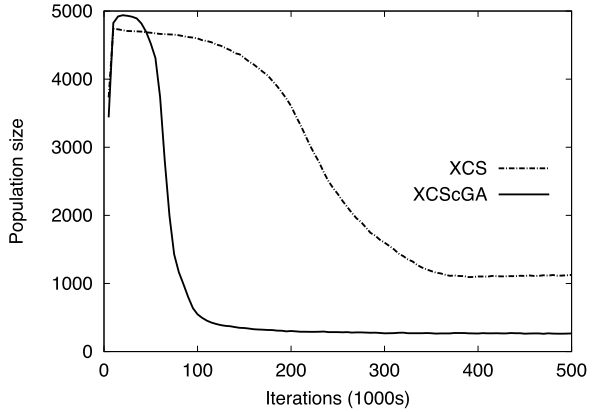

図 8　37-Multiplexer 問題における XCS と XCScGA の分類子数
Fig. 8　Population sizes of XCS and XCScGA on 37-Multiplexer problem.

的探索を行った後，[P] 内の不要な分類子の数が減少していることを意味する．また，学習終了後の XCS の分類子数は 1,125 であるのに対し，XCScGA の分類子数は 267 であることから，XCScGA は XCS の分類子数を約 76.3%削減していることが分かる．

6.4　考察

Multiplexer 問題の実験結果から，XCScGA は XCS よりも 1) 少ない学習回数で正答率が 1 に達し，2) 少ない分類子数で学習可能であることが明らかになった．この要因として，XCScGA が用いる cGA を用いた分類子生成法は，XCS が用いる GA を用いた分類子生成法よりも，a) 少ない進化回数で正しい部分解を持つ分類子を生成可能であり，b) 不要な分類子の生成を抑制可能である点があげられる．

一般的に正答率が上昇するためには，正しい部分解を持つ分類子が生成されなければならない．XCS は正答率が 1 に達するまでに多くの学習回数を要することから，GA を用いた分類子生成法は，正しい部分解を持つ分類子を生成するまでに多大な進化回数が必要であることが分かる．そのため同分類子生成法は，進化回数の増加にともなって不要な分類子を多数生成する．一方で，XCScGA は少ない学習回数で正答率が 1 に達していることから，cGA を用いた分類子生成法は，少ない進化回数で正しい部分解を持つ分類子を生成可能であることが分かる．加えて，同分類子生成法は正しい部分解を持つ分類子を生成後，確率モデル内の確率が収束するため，正しい部分解を持つ分類子と同様の子個体を生成する傾向がある．その結果，不要な分類子の生成が抑制され，分類子数が減少したと考えられる．

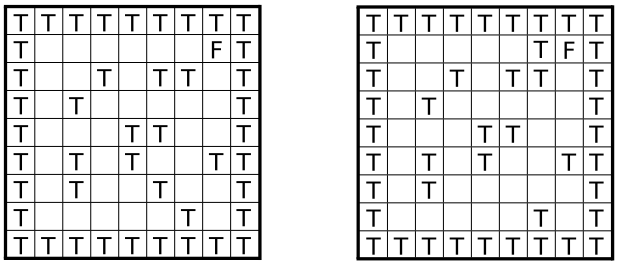

図 9　フィールド（maze5，maze6）
Fig. 9　maze5 and maze6.

7.　強化学習問題における実験

本章では，強化学習問題における実験を通して，提案手法の有効性を評価する．具体的には，強化学習問題における LCS の一般的なベンチマーク問題として，Grid world 問題 [7], [38] を用いて実験を行う．

7.1　Grid world 問題

Grid world 問題は難易度の異なるフィールド（maze）において，*animat* と呼ばれるエージェントが各セル間を遷移し食料（報酬）の獲得を目的とする問題である．フィールドは障害物（Obstacle: “T”），通路（Empty position: “ ”），食料（Food: “F”）で構成される．また，とりうる行動の種類は 8 種類であり，現在位置から 8 近傍の各状態へ移動できる．

各セルは障害物が 01，通路が 00，食料が 11 でそれぞれ表す．状態を表すコードは，現在位置を中心とする 8 近傍のセルを用いて決定される．具体的には，現在位置を中心として真上のセルから時計回りにコーディングされ，長さ 16 のビットで与えられる．ここで 10 は意味をなさない．まず，エージェントはフィールド内の通路にランダムに配置される．障害物の方向へ移動する場合は，通行不可として移動前の状態にとどまる．エージェントが食料に達した場合，報酬 $r = 1{,}000$ を獲得し探索が終了する．ただし探索ステップ数が最大ステップ数 $maxstep$ を越えた場合も探索を終了するが報酬は与えられない．

本論文では**図 9** に示す 2 種類の異なるフィールド（maze5 と maze6）[21] を用いる．maze6 は maze5 と類似したフィー

ルドを持つが，食料を獲得する経路が 1 つであることから，maze5 よりも難しい問題とされる [7]．また，maze5 と maze6 における最短平均経路（optimum step）はそれぞれ 4.61，5.19 である [5], [7]．

7.2　実験設定

7.2.1　概要

提案手法を導入した XCS（XCScGA）と XCS を Grid world 問題に適用し，後述する評価基準を用いて提案手法の有効性を評価する．また，Multiplexer 問題における実験設定と同様に，実験は学習段階と評価段階から構成され，交互に実行する．学習段階ではランダム行動選択を行うが，評価段階では greedy 行動選択を行い，発見部は適用しない．

7.2.2　評価基準とパラメータ設定

評価基準は，1) 食料までのステップ数（Performance），2) 分類子集団 [P] 内の分類子数（Population size）を用いる．食料までのステップ数は小さく最短平均経路に近いほど，正確に一般化された分類子を学習できていることを示している．[P] 内の分類子数は少ない値であるほど，分類子集団中に不要な分類子が少ないことを意味する．食料までのステップ数は，評価段階でフィールド内の通路にランダムに配置されたエージェントが，greedy 行動選択したときのステップ数から計算する．ただし，ステップ数が最大ステップ数 $maxstep$ を越えた場合，食料までのステップ数を $maxstep$ に設定し，評価を終了する．分類子数は，評価段階で，エージェントが問題終了条件（食料を獲得するか最大ステップ数を越えた場合）に達したときの分類子集団内の分類子数を設定する．学習回数は 1 試行につき，maze5 ならびに maze6 ともに 3,000 回行う．このとき，各評価基準は学習回数 50 回ごとの移動平均で示し，試行数 10 回の平均をとる．

パラメータ設定は先行研究と同様に設定する [7]．具体的には, maze5 における XCS のパラメータ設定は, $N = 3{,}000$, $\nu = 5$, $\beta = 0.2$, $\epsilon_0 = 10$, $\theta_{mna} = 8$, $\theta_{GA} = 25$, $\chi = 0.8$, $\mu = 0.01$, $P_{\#} = 0.3$, $\tau = 0.4$ とするが，$maxstep = 50$ と設定する [5]．XCScGA は，$sr = 0.2$（20%），$P_C = 0.2$ と設定し，その他のパラメータは XCS と同様に設定する．maze6 における XCS と XCScGA のパラメータ設定は，$\epsilon_0 = 1$，$\theta_{GA} = 100$ と設定し，その他は maze5 と同様である．

7.3　実験結果

7.3.1　maze5

図 10，**図 11** に，maze5 における XCS と XCScGA の食料までのステップ数ならびに [P] 内の分類子数を示す．各図の縦軸は，それぞれ食料までのステップ数（Performance），分類子数（Population size）であり，横軸は学習回数（Iterations）を示している．

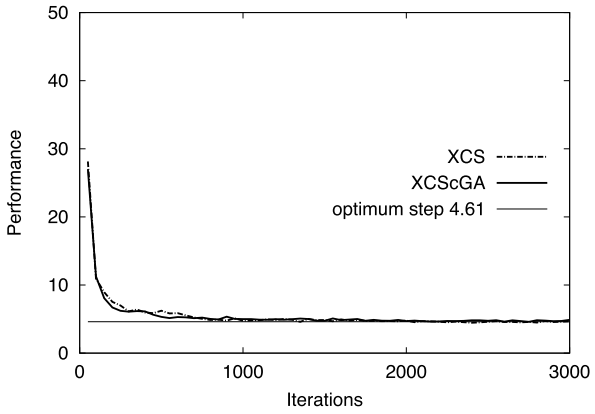

図 10　maze5 における XCS と XCScGA の食料までのステップ数
Fig. 10　Performance of XCS and XCScGA on maze5.

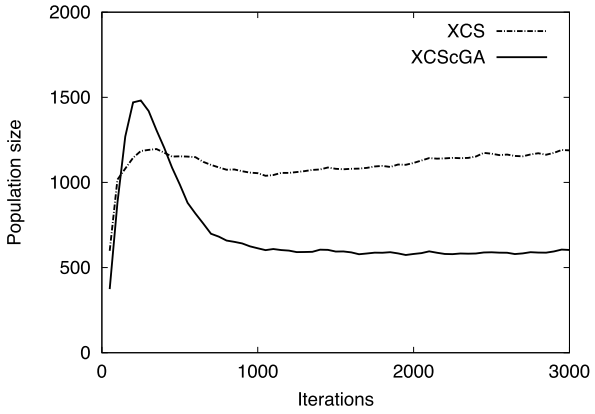

図 11　maze5 における XCS と XCScGA の分類子数
Fig. 11　Population size of XCS and XCScGA on maze5.

両図より，XCS と XCScGA のステップ数は，同程度の学習回数で最短平均経路（optimum step）に収束していることから，正しい分類子を適切に学習していることが分かる．XCS の分類子数は，学習回数が経過しても分類子数が増加している．これは，XCS が不要な分類子を継続して生成していることを示している．

一方で，XCScGA の分類子数は，学習回数 150〜450 回の間は XCS よりも大きい値であるが，その後分類子数が次第に減少している．これは，XCScGA が，学習初期に分類子の大域的探索を行った後，分類子集団中の不要な分類子の数が減少していることを示している．その結果，学習終了後の XCS の分類子数は 1,188 であるのに対し，XCScGA の分類子数は 603 であることから，XCScGA は XCS の分類子数を約 49.2%削減していることが分かる．

7.3.2　maze6

図 12，**図 13** に，maze6 における XCS と XCScGA の食料までのステップ数ならびに [P] 内の分類子数を示す．

両図より，maze5 の実験結果と同様に，XCS と XCScGA のステップ数は，同程度の学習回数で最短平均経路（optimum step）に収束していることから，正しい分類子を

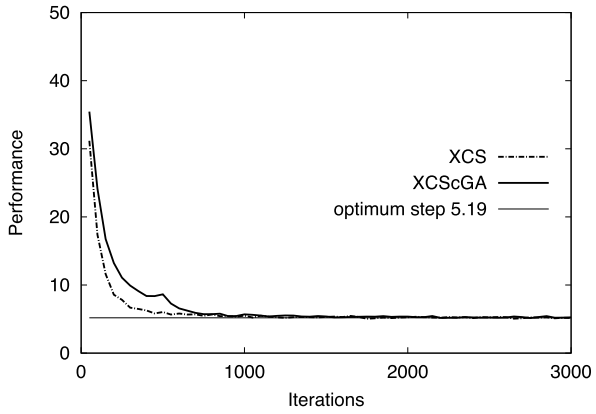

図 12　maze6 における XCS と XCScGA の食料までのステップ数
Fig. 12　Performance of XCS and XCScGA on maze6.

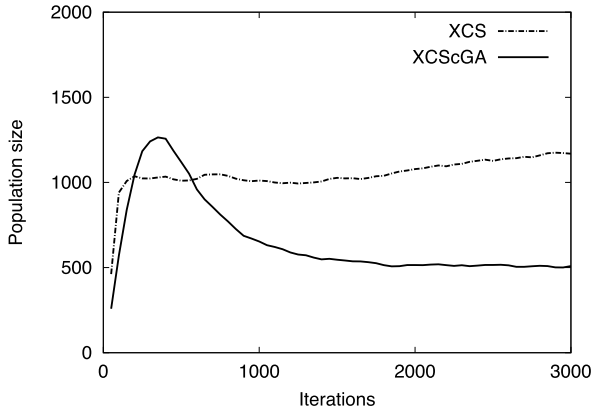

図 13　maze6 における XCS と XCScGA の分類子数
Fig. 13　Population size of XCS and XCScGA on maze6.

適切に学習していることが分かる．XCS の分類子数は，学習回数が経過しても分類子数が増加している．一方で，XCScGA の分類子数は，学習回数 250〜500 回の間は XCS よりも大きい値であるが，その後分類子数が減少している．また，学習終了後の XCS の分類子数は 1,288 であるのに対し，XCScGA の分類子数は 508 であり，XCScGA は XCS の分類子数を約 54.7%削減していることが分かる．

7.4　考察

Grid world 問題の実験結果から，XCScGA は XCS よりも少ない分類子数で学習可能であることが明らかになった．この要因として，XCScGA が用いる cGA を用いた分類子生成法は，XCS が用いる GA を用いた分類子生成法よりも，特に不要な分類子の生成を抑制可能である点があげられる．

Grid world 問題では，とりうる行動が 8 種類存在するため，正しい分類則（IF，THEN の組合せ）を持たない不要な分類子が多数存在する．そのため，XCS を Grid world 問題に適用した場合，GA を用いた分類子生成法により多数の不要な分類子が生成され，分類子数が増加する．一方で XCScGA では，第 6 章の考察と同様に，cGA を用いた分類子生成法は，確率モデル内の確率が収束後，不要な分類子の生成を抑制可能である．そのため，特に Grid world 問題では，XCS とは対照的に分類子数が削減したことが考えられる．

8.　追加実験

本章では，提案手法のメカニズムの効果ならびに新たに導入したパラメータの影響を検証するために，追加実験を行う．具体的には，まず，1) 提案手法が不要な分類子の生成を抑制する効果について検証する．次に，2) パラメータ sr と P_C が提案手法に与える影響について分析する．

8.1　不要な分類子の生成を抑制する効果の検証

提案手法は，確率ベクトルを用いて有望な部分解を抽出し，有望でない部分解を持つ子個体（不要な分類子）の生成を抑制することで，分類子数を削減することを目的としている．ここでは，その不要な分類子の生成を抑制する効果を検証する．そのために，Multiplexer 問題上の正しい部分解と子個体の部分解を比較する．具体的には，20-Multiplexer 問題において，生成された子個体の属性値（generalized bit #もしくは specified bit 0，1）を評価する．

8.1.1　評価方法

まず，20-Multiplexer 問題上の正しい部分解について述べる．一般的に，Multiplexer 問題では正確かつ最も一般化された分類子（最適な分類子）が一意に定まる．たとえば，6-Multiplexer 問題（$k=2$，$l=6$）における最適な分類子は，11###1 や 01#1###等の条件部を持つ．つまり，最適な分類子の条件部 C は，C_0〜C_{k-1} と出力に対応する C_{k+d} は必ず specified bit（0，1）となり，出力に対応しないその他の C_k〜C_{l-1} は generalized bit（#）となる．以上より，20-Multiplexer 問題（$k=4$，$l=20$）における正しい部分解は，C_0〜C_3 が specified bit であり，C_4〜C_{19} は，そのほとんどが generalized bit（#）となる．

次に，評価方法として，毎世代（Iterations）に生成された子個体の各遺伝子座 C_0〜C_{19} について，specified bit である割合（Specificity）を計算する．Specificity の値が 1 であれば，その遺伝子座は specified bit であり，0 であれば generalized bit であることをそれぞれ意味する．具体的には，毎世代の子個体の Specificity は，生成された 2 つの子個体の属性値から算出する．たとえば，2 つの子個体の条件部が 1####11 と 1#1##11 であるとき，C_0 の specificity は 1，C_1 は 0，C_2 は 0.5 となる．実際には，同様の属性値を持つべき遺伝子座である C_{0-3} の Specificity の平均値 Specificity(C_{0-3}) と，C_{4-19} における平均値 Specificity(C_{4-19}) をそれぞれ算出し，XCS と XCScGA の Specificity を比較する．また，Specificity は 20-Multiplexer 問題に適用時に生成された子個体を用いて評価する．その際の実験設定は，6 章と同様であり，Specificity は学習回

数 5,000 回の移動平均で示し，10 試行の平均をとる．

8.1.2　評価方法

図 14 に，20-Multiplexer 問題における XCS と XCScGA の Specificity を示す．図の縦軸は，それぞれ遺伝子座が specified bit である割合（Specificity）であり，横軸は学習回数 (Iterations) を示している．ここで，C_{0-3} は specified bit となるべきであるため，specificity(C_{0-3}) は 1 付近に収束し，同様に C_{4-19} は generalized bit となるべきであるため，specificity(C_{4-19}) は 0 に近い値に収束することが好ましい．ただし，C_{4-19} は specified bit となるべき C_{k+d} も含まれるため，実際は 0 よりも大きい値をとる．

図より，XCScGA における Specificity(C_{0-3}) は，XCS より高い値であり 1 付近に収束している（子個体には突然変異が適用されるため，完全に 1 に収束しない）．一方で，Specificity(C_{4-19}) についても，XCS よりも低い値を持つ．これは，XCS は正しい部分解を持つ分類子だけでなく，正しい部分解を持たない不要な分類子についても生成し続けていることを意味する．一方で，XCScGA は，正しい部分解を持つ分類子を XCS よりも多数生成し，不要な分類子を生成する数が XCS よりも少ないことが分かる．したがって，提案手法は，不要な分類子の生成を抑制する効果があることが分かる．

8.2　パラメータ sr と P_C の分析

ここでは，提案手法に新たに導入した 2 つのパラメータである 1) 選択する親個体数を決定するパラメータ sr と 2) 確率ベクトルに基づく突然変異の適用確率を決定するパラメータ P_C の影響を分析する．具体的には，20-Multiplexer 問題において，それぞれのパラメータの値を変化させたときの正答率ならびに $[P]$ 内の分類子数について比較する．実験設定は 6 章と同様である．

8.2.1　パラメータ sr の分析

図 15 と **図 16** にパラメータ $sr = 0.05, 0.1, 0.2, 0.5, 1.0$ と設定したときの 20-Multiplexer 問題における正答率と $[P]$ 内の分類子数を示す．ここで，sr が小さい値になるほど，親個体の数が少なくなることを意味する．

図より，$sr = 0.2, 0.5$ 設定時では，最も速く（少ない学習回数で）正答率が 1 に達しているが，学習回数 10,000 回では，$sr = 0.2$ 設定時では，$sr = 0.5$ 設定時よりも正答率が高い．$sr = 0.1$ と 0.05 に設定した場合は，sr が小さい値になるに従って，次第に学習速度が低下していることが分かる．さらに，$sr = 1.0$ 設定時では，$sr = 0.2, 0.5$ 設定時よりも学習速度が低下している．また分類子数については，sr の値が小さくなるに従って，学習初期（学習回数 5,000〜10,000 回）での分類子数が次第に減少していることが分かる．加えて，$sr = 0.5$ 設定時では，学習回数 20,000 回から 30,000 回にかけて最も速く分類子数が減少している．

パラメータ sr の値により正答率が 1 に達するまでの学習回数に差が生じた理由としては，学習効率が分類子数に依存することが原因である．一般的に，学習初期では，多様な分類子を生成することが正しい分類子の獲得に寄与す

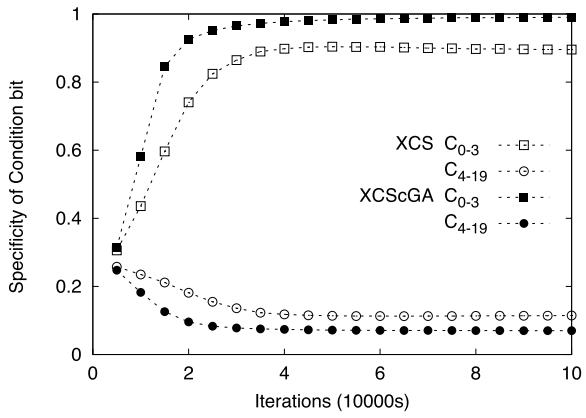

図 14　20-Multiplexer 問題における XCS と XCScGA の Specificity の比較

Fig. 14　Specificities of XCS and XCScGA on 20-Mulitiplexer.

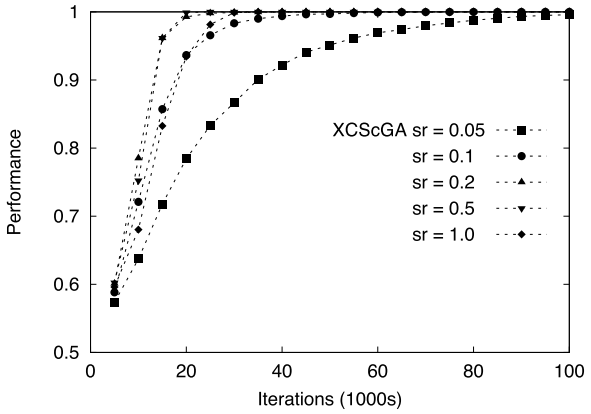

図 15　20-Multiplexer 問題におけるパラメータ sr を変化させたときの正答率

Fig. 15　Performances of XCScGA with different values of parameter sr.

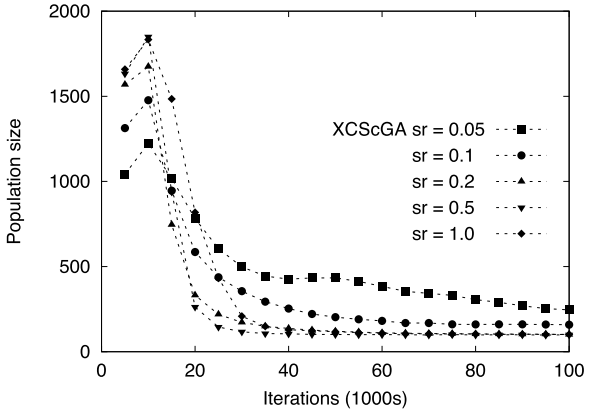

図 16　20-Multiplexer 問題におけるパラメータ sr を変化させたときの分類子数

Fig. 16　Population sizes of XCScGA with different values of parameter sr.

るが，過剰に分類子を生成した場合，不要な分類子の存在により学習効率が低下する [8]．したがって，$sr = 1.0$ 設定時では，学習回数 15,000 回においても分類子数が多く，不要な分類子を学習することで，学習効率が低下する．そのため，正答率が 1 に達するまでに多くの学習回数を要する．一方で，学習初期に分類子数が少ない場合，分類子の多様性が減少し，正しい分類子を生成するまでに多くの世代数（学習回数）を要する．そのため，分類子数が少ない $sr = 0.05, 0.1$ 設定時では，正答率が 1 に達するまでに多くの学習回数を要する．このような理由から，学習初期では適度な分類子数である $sr = 0.2, 0.5$ 設定時は，少ない学習回数で正答率が 1 に達する．

また，パラメータ sr の値により分類子数が変化する理由としては，親個体の数が確率ベクトル内の確率の値に影響することが原因である．具体的には，sr が小さい値であり親個体の数が少ない場合，$[S]$ 内に存在する部分解の種類数が少なく，各確率は 1 もしくは 0 に収束しやすい．一方で，sr が大きい値であり親個体の数が多い場合，多種類の部分解が存在するため，各確率は 1 もしくは 0 に収束しない傾向がある．そのため，sr が小さい値であれば，同一の部分解を持つ子個体を生成することで分類子数が増加しない．また，sr が大きい値であれば，確率が収束しないため多様な子個体を生成し分類子数は増加する．

以上より，パラメータ sr は，生成する分類子の多様性に影響することで，学習速度を決定する要因であることが分かる．加えて，学習状況に応じて，最も高い正答率と最も少ない分類子数を実現する sr の値が異なることから，学習状況を考慮した sr の適応的制御が課題である．

8.2.2　パラメータ P_C の分析

図 17 と **図 18** にパラメータ $P_C = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0$ と設定したときの 20-Multiplexer 問題における正答率と $[P]$ 内の分類子数を示す．ここで，P_C が小さい値になるほど，確率ベクトル型突然変異を実行する確率が高くなることを意味する．つまり，$P_C = 0.0$ であれば，確率ベクトル型突然変異をつねに実行するが，$P_C = 1.0$ であれば，通常の突然変異（Niche mutation）をつねに実行することを意味する．

図より，XCScGA の正答率は，P_C の値が小さくなるに従って，学習初期（学習回数 5,000〜15,000）の正答率が高くなる傾向がある．しかし，$P_C = 0.0$ では，正答率が完全に 1 に収束していない．また，XCScGA の分類数は，P_C の値が小さくなるに従って，分類数が減少していることが分かる．

パラメータ P_C の値の変化が正答率と分類子数に影響する理由としては，突然変異が分類子の多様性に影響することが原因である．通常の突然変異（Niche mutation）は，新しい部分解を持つ分類子を生成することで，正しい分類子を探索することを目的としている．一方で，確率ベクトルに基づく突然変異は，分類子の条件部を確率モデルで示唆された部分解に修正することを目的としている．そのため，通常の突然変異は多様な分類子を生成する（分類子の多様性が高くなる）が，確率ベクトルに基づく突然変異は類似する分類子を生成する（分類子の多様性が低くなる）点で異なる．そのため，$P_C = 0.0$ 設定時では，通常の突然変異を実行しないため，新しい部分解を持つ分類子の生成が促進されず，正しい分類子を生成できないことで正答率は 1 に完全に収束していない．一方で，P_C を高い値に設定するに従って，通常の突然変異が頻繁に実行され，新しい部分解を持つ分類子が生成されるが，不要な分類子の生成を促すことになり学習効率が低下する．以上より，パラメータ P_C は，パラメータ sr と同様に分類子の多様性に影響することで，学習速度と特に分類子数に影響することが分かる．

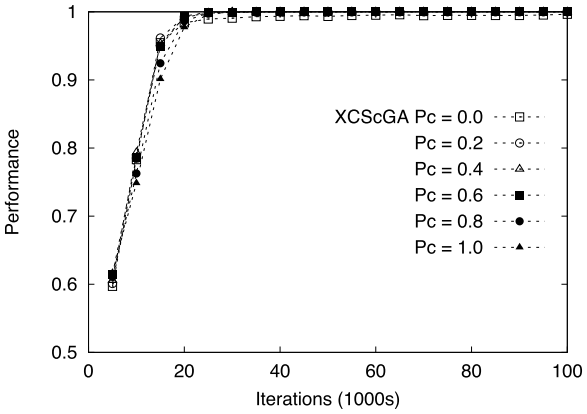

図 17　20-Multiplexer 問題におけるパラメータ P_C を変化させたときの正答率

Fig. 17　Performances of XCScGA with different values of parameter P_C.

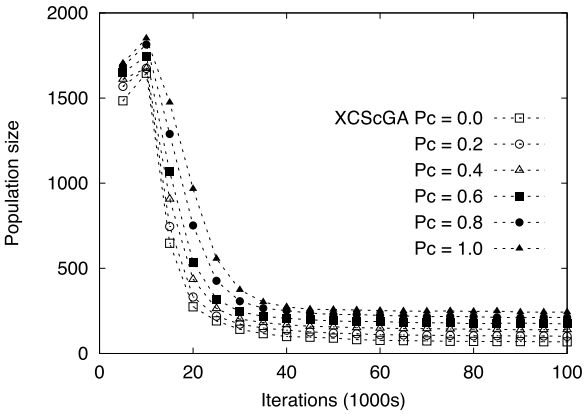

図 18　20-Multiplexer 問題におけるパラメータ P_C を変化させたときの分類子数

Fig. 18　Population sizes of XCScGA with different values of parameter P_C.

9.　おわりに

本論文では，現在主流の学習分類子システム（XCS）が膨

大な分類子数を要するという問題に対し，Compact Genetic Algorithm を用いた確率モデル型分類子生成法を提案した．提案分類子生成法は，分類子が持つ部分解の存在確率をモデル化することで不要な分類子の生成を抑制し，XCS が必要な分類子数を削減する．提案手法の特徴は，1) 従来の分類子数削減手法と比較して，学習段階中に分類子数の削減が可能であり，2) 従来の確率モデル型分類子生成法が適用困難であった強化学習問題クラスに適用可能である．提案手法の有効性を検証するために，提案法を導入した XCS（XCScGA）を教師あり学習問題（Multiplexer 問題）と強化学習問題（Grid world 問題）に適用したところ，次の知見を得た．まず，1) 提案 LCS は従来 LCS よりも少ない学習回数で最適解を学習可能であり，2) 従来 LCS が学習した分類子数に対し，提案 LCS は最小でも 49%（最大で 76%）削減した分類子数で学習可能であることを示した．

今後の課題としては，実環境問題の適用に向けて，次の取り組むべき課題があげられる．まず，1) 実数値環境に適用するために提案手法を拡張する．たとえば，確率ベクトルを用いた EDA である Population-Based Incremental Learning（PBIL）[2] を実数値に拡張した Real-code PBIL [26] を用いることが考えられる．また，2) 雑音を含む問題においても正しい部分解を抽出するために，提案手法に遺伝子座間の依存関係を考慮した確率モデルを導入する．最後に，3) パラメータ *sr* を学習状況に応じて適応的に制御することで，分類子数の多様性を制御可能な分類子生成法を構築する．

参考文献

[1] Bacardit, J. and Butz, M.V.: *Data Mining in Learning Classifier Systems: Comparing XCS with GAssist*, IlliGAL Report No.2004030 (2004).

[2] Baluja, S.: Population-based incremental learning. A method for integrating genetic search based function optimization and competitive learning, Technical report, DTIC Document (1994).

[3] Bernadó-Mansilla, E. and Garrell-Guij, M.J.: Accuracy-based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks, *Evolutionary Computation*, Vol.11, pp.209–238 (2003).

[4] Bull, L., Bernadó-Mansilla, E. and Holmes, J.H. (eds.): *Learning Classifier Systems in Data Mining*, Studies in Computational Intelligence, Vol.125, Springer (2008).

[5] Butz, M.V.: *Rule-Based Evolutionary Online Learning Systems*, Springer (2006).

[6] Butz, M.V., Goldberg, D.E. and Tharakunnel, K.: Analysis and Improvement of Fitness Exploitation in XCS: Bounding Models, Tournament Selection, and Bilateral Accuracy, *Evolutionary Computation*, Vol.11, No.3, pp.239–277 (2003).

[7] Butz, M.V., Goldberg, D. and Lanzi, P.L.: Gradient Descent Methods in Learning Classifier Systems: Improving XCS Performance in Multistep Problems, *Evolutionary Computation*, Vol.9, No.5, pp.452–473 (2005).

[8] Butz, M.V., Kovacs, T., Lanzi, P.L. and Wilson, S.W.: Toward a Theory of Generalization and Learning in XCS, *IEEE Trans. Evolutionary Computation*, Vol.8, No.1, pp.28–46 (2004).

[9] Butz, M.V. and Wilson, S.W.: An Algorithmic Description of XCS, *Journal of Soft Computing*, Vol.6, No.3–4, pp.144–153 (2002).

[10] Dam, H.H., Abbass, H.A. and Lokan, C.: DXCS: An XCS System For Distributed Data Mining, *Proc. Genetic and Evolutionary Computation Conference* (*GECCO2005*), pp.1883–1890 (2005).

[11] Ebadi, T., Zhang, M. and Browne, W.: XCS-based versus UCS-based feature pattern classification system, *Proc. 14th International Conference on Genetic and Evolutionary Computation Conference*, pp.839–846, ACM (2012).

[12] Fu, C. and Davis, L.: A Modified Classifier System Compaction Algorithm, *GECCO*, Vol.2, pp.920–925 (2002).

[13] Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley (1989).

[14] Harik, G.R., Lobo, F.G. and Goldberg, D.E.: The compact genetic algorithm, *IEEE Trans. Evolutionary Computation*, Vol.3, No.4, pp.287–297 (1999).

[15] Holland, J.H.: Escaping Brittleness: The Possibilities of General Purpose Learning Algorithms Applied to Parallel Rule-based System, *Machine Learning*, Vol.2, pp.593–623 (1986).

[16] Hurst, J., Bull, L. and Melhuish, C.: TCS Learning Classifier System Controller on a Real Robot, *Proc. 7th International Conference on Parallel Problem Solving from Nature*, *PPSN VII*, pp.588–600, Springer-Verlag (2002).

[17] Katagami, D. and Yamada, S.: Active teaching for an interactive learning robot, *Proc. 12th IEEE International Workshop on Robot and Human Interactive Communication*, *ROMAN 2003*, pp.181–186 (2003).

[18] Kharbat, F., Bull, L. and Odeh, M.: Mining breast cancer data with XCS, *Proc. Genetic and Evolutionary Computation Conference* (*GECCO2007*), pp.2066–2073 (2007).

[19] Kovacs, T.: Genetics-based Machine Learning, *Handbook of Natural Computing: Theory, Experiments, and Applications*, Rozenberg, G., Bäck, T. and Kok, J. (eds.), Springer-Verlag (2011).

[20] Kovacs, T.: Strength or Accuracy? Fitness Calculation in Learning Classifier Systems, *Learning Classifier Systems. From Foundations to Applications*, Vol.1813, pp.143–160 (2000).

[21] Lanzi, P.L.: An Analysis of Generalization in the XCS Classifier System, *Evolutionary Computation Journal*, Vol.7, No.2, pp.125–149 (1999).

[22] Larrañaga, P. and Lozano, J.: *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Academic Publisher (2001).

[23] Llora, X., Sastry, K. and Goldberg, D.: The compact classifier system: Scalability analysis and first results, *2005 IEEE Congress on Evolutionary Computation 2005*, pp.596–603 (2005).

[24] Mühlenbein, H. and Paaß, G.: From recombination of genes to the estimation of distributions I. binary parameters, *Parallel Problem Solving from Nature*, pp.178–187, Springer-Verlag (1996).

[25] Ravichandran, B., Gandhe, A. and Smith, R.E.: XCS for robust automatic target recognition, *Proc. 2005 Conference on Genetic and Evolutionary Computation*, pp.1803–1810, ACM (2005).

[26] Sebag, M. and Ducoulombier, A.: Extending population-based incremental learning to continuous search spaces, *Parallel Problem Solving from Nature — PPSN V*, pp.418–427, Springer (1998).
[27] Shi, L., Shi, Y. and Gao, Y.: Clustering with XCS and agglomerative rule merging, *Intelligent Data Engineering and Automated Learning — IDEAL 2009*, pp.242–250, Springer (2009).
[28] Smith, S.F.: A Learning System Based on Genetic Adaptive Algorithms, *Ph.D. Dissertation*, University of Pittsburgh, Pittsburgh (1980).
[29] Sutton, R.S.: Learning to Predict by the Methods of Temporal Differences, *Machine Learning*, Vol.3, No.1, pp.9–44 (1988).
[30] Sutton, R.S. and Barto, A.G.: *Reinforcement Learning — An Introduction*, MIT Press (1998).
[31] Tamee, K., Bull, L. and Pinngern, O.: Towards clustering with XCS, *Proc. 9th Annual Conference on Genetic and Evolutionary Computation*, pp.1854–1860, ACM (2007).
[32] Tzima, F.A. and Mitkas, P.A.: ZCS Revisited: Zeroth-Level Classifier Systems for Data Mining, *IEEE International Conference on Data Mining Workshops* (*ICDMW2008*), pp.700–709 (2008).
[33] Tzima, F.A. and Mitkas, P.A.: Strength-based learning classifier systems revisited: Effective rule evolution in supervised classification tasks, *Engineering Applications of Artificial Intelligence* (2012).
[34] Urbanowicz, R.J., Granizo-Mackenzie, A. and Moore, J.H.: Instance-linked attribute tracking and feedback for Michigan-style supervised learning classifier systems, *GECCO*, pp.927–934 (2012).
[35] Urbanowicz, R.J., Andrew, A.S., Karagas, M.R. and Moore, J.H.: Role of genetic heterogeneity and epistasis in bladder cancer susceptibility and outcome: A learning classifier system approach, *Journal of the American Medical Informatics Association*, Vol.20, No.4, pp.603–612 (2013).
[36] Whitley, D. and Kauth, J.: *GENITOR: A different genetic algorithm*, Colorado State University, Department of Computer Science (1988).
[37] Wilson, S.W.: ZCS: A Zeroth Level Classifier System, *Evolutionary Computation*, Vol.2, No.1, pp.1–18 (1994).
[38] Wilson, S.W.: Classifier fitness based on accuracy, *Evolutionary Computation*, Vol.3, No.2, pp.149–175 (1995).
[39] Wilson, S.W.: Mining Oblique Data with XCS, *Revised Papers from the 3rd International Workshop on Advances in Learning Classifier Systems*, *IWLCS '00*, pp.158–176, Springer-Verlag (2001).
[40] Zang, Z., Li, D. and Wang, J.: Knowledge extraction and rule set compaction in XCS for non-Markov multi-step problems, *Evolutionary Intelligence*, Vol.6, No.1, pp.41–53 (2013).

中田 雅也

1988 年生．2013 年電気通信大学大学院情報理工学研究科総合情報学専攻修士課程修了（工学）．同年，同大学院同研究科同専攻博士課程入学，現在に至る．2012 年 2 月～5 月 Politecnico di Milano，2013 年 11 月～2014 年 9 月 University of Bristol，2014 年 9 月～現在 Victoria University of Wellington 各 Research visiting student．IEEE CIS Japan Chapter Young Researcher Award，最優秀発表賞（進化計算学会シンポジウム 2012），システム制御学会奨励賞各受賞．International Workshop on Learning Classifier Systems（IWLCS）2015–2016 Co-organizer．進化計算，学習分類子システム，データマイニング等の研究に従事．IEEE，ACM，計測自動制御学会，システム制御情報処理学会，進化計算学会各会員．日本学術振興会特別研究員（DC1）．

Pier Luca Lanzi

Pier Luca Lanzi received his Ph.D. degree in computer and automation engineering from the Politecnico di Milano, Milano, Italy. He is an Associate Professor at the Department of Electronics, Information and Bioinformatics, Politecnico di Milano. His research areas include evolutionary computation, reinforcement learning, and machine learning. He is interested in applications to data mining and computer games. He is a member of the editorial board of the Evolutionary Computation Journal, the IEEE Transactions on Computational Intelligence and AI in Games, and Evolutionary Intelligence. He is also the Editor-in-Chief of the SIGEVOlution, the newsletter of the ACM Special Interest Group on Genetic and Evolutionary Computation.

田島 友祐

1989 年生．2013 年電気通信大学電気通信学部人間コミュニケーション学科卒業．同年，同大学大学院情報理工学研究科総合情報学専攻修士課程入学，現在に至る．進化計算の研究に従事．進化計算学会会員．

高玉 圭樹　（正会員）

1970 年生．1998 年東京大学大学院工学系研究科博士課程修了．博士（工学）．同年，国際電気通信基礎技術研究所（ATR）入所．2002 年東京工業大学大学院総合理工学研究科講師を経て，2006 年電気通信大学電気通信学部助教授，2007 年准教授，2011 年教授，現在に至る．マルチエージェントシステム，強化学習，創発的計算手法の研究に従事．著書に『マルチエージェント学習—相互作用の謎に迫る—』（コロナ社）等．IEEE，人工知能学会，日本ロボット学会，電子情報通信学会等各会員．